Computational Thinking

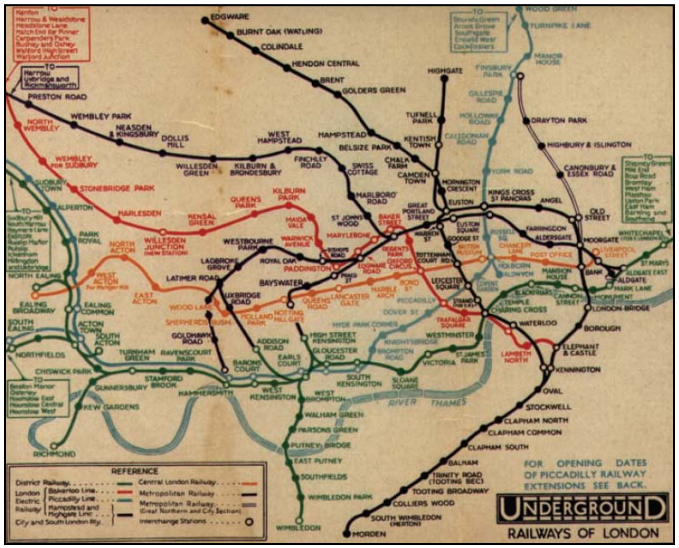
Abstraction

Abstraction

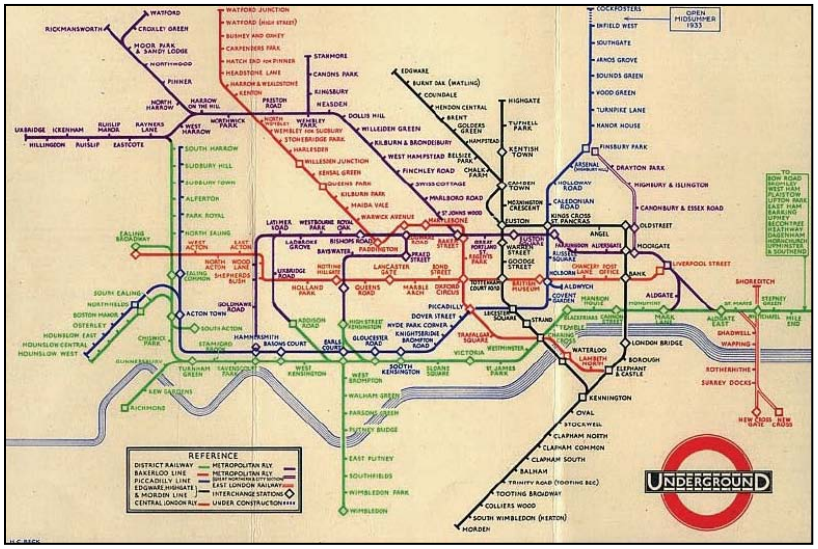
- Goals
  - What is abstraction?
  - Is it teachable?
  - How to assess?
- Abstraction has two facets
  - “Removing detail to simplify and focus attention” [p38]
  - “identifying the common core or essence” [p38]
- Cautions
  - Level of detail has to be carefully selected
  - “The level, benefit, and value of a particular abstraction depend on its purpose. …misleading if used for other purposes” [p39]

Unless otherwise noted references from here forward are to [Kramer 2007].

Example



1928 map of London underground system



Beck's 1931 map of London underground system

Importance

- "Abstraction skills are essential in the construction of appropriate models, designs, and implementations that are fit for the particular purpose at hand." [p40]
- Analysis
  - Requirement elicitation
  - "identifying the critical aspects of the environment"
- Synthesis
  - Design
  - Avoid unnecessary implementation constraints
- Managing complexity
  - By setting aside non-essential details
  - Through layers of abstraction

Teaching Abstraction

- Author's institution does not teach abstraction separately
  - " abstraction is an essential aspect of computing, but that it must be taught indirectly through other topics." [p41]
- Math helps
  - Citing Devlin: "The main benefit of learning and doing mathematics is not the specific content; rather it's the fact that it develops the ability to reason precisely and analytically about formally defined abstract structures" [p41]
- Has some presence in ACM software engineering curriculum
- Can be practiced by formal modeling and analysis
- Student motivation
  - Enhanced by problem-oriented approach
  - Benefit from tool support

Assessment

- Gather data by
  - "Measur[ing] students abstraction abilities annually while at college." [p42]
    - Determine correlation with other measures of learning
    - Provide an additional means of assessment
  - Measur[ing] students abstraction abilities at the time of application to study computing." [p42]
    - Select students with most suitable skills, not just those academically qualified
- Assessment tests
  - Needed but unavailable
  - Proposal (from Hazzan) for tests with
    - Different kinds of tasks and descriptions
    - Quantitative and qualitative data
    - Open-ended questions and interviews

Criticism of abstraction

- Thesis
  - "abstract formal descriptions…might in fact be antagonistic to reasonable human concerns." [p8]
  - "is important to be alert to potential trade-offs inherent in the advocacy and adoption of one particular style of thinking and problem-solving." [p8]
- Abstraction misfits
  - "A misfit is a correspondence problem between abstractions in the device, abstractions in the shared representation (the user interface) and abstractions as the user thinks about them." [p3]
- Side-effects of CT
  - Literalist thinking: "only capable of manipulating the explicitly available syntax and mathematically-structured 'semantics' of information, not its socially constructed counterpart. " [p6]
  - Goal conflation: "whether a system is successful is no longer measured by its actual efficacy, but rather by some property of its abstract structure. " [p6]
  - False support: "abstract descriptions, by rising above the circumstances of any specific instance, offer an illusion of universality and universal support. In this respect, computational thinking can become a misleading foundation for scientific work, through encouraging an abstract 'laboratory' that is not founded in any real or human phenomenon." [p7]

Unless otherwise noted references from here forward are to [BCG 2008].

Abstraction and Users

- "Unfortunately, clean technological models of the world generally do not offer a good fit to human ones." [p4]
- Steps to abstracting away inconvenient complexity [p4-5]
  - Abstract away the user:
    - "becoming a somewhat simplified version of what the real user is"
    - Once "all users [are] represented in a single form, they can be aggregated
      - May be too inclusive (unable to differentiate significant variation)
      - "user blindness" (harder to reason about what is important to a user)
      - Confused discourse (physical person vs. abstract 'user')
  - Dehumanise the User
    - "eliding aspects of people that the people may perceive as being important to their humanity"
    - Loss of rich social context (e.g., enforcing security practices)
  - Change the user
    - "problematic abstractions of the system may appear in the user interface"
    - "to maintain the integrity of the computational model, it is [sometimes] necessary to be assertive in the way the technology is deployed."
    - Dehumanized abstractions may make some things unknowable

Design conflicts

- "best practices in user centered design tend to be grounded in the needs and actions of specific users. The design of software architectures and organisational processes, in contrast, aims to identify, implement and maintain effective data and process abstractions." [p2]
- "…an observed tension between abstract descriptions of the system, and descriptions of specific cases. In some cases, users offered abstract descriptions based on generalisation over their own repeated experience, but these conflicted with the abstractions used by the system developers." [p2]
- Examples
  - A graduate student user passed through several menu levels each with only one choice
    - Design "correct" from the developer point of view
    - Design "wrong" from the user point of view
  - Research accounting processes were incompatible with the actual conduct of the research
    - "That structure represented an abstract conceptualisation of the process of doing research; the specific experiences of system users inconveniently crossed categories and thus defied classification according to the 'correct' accounting abstractions. "

Implications for Design

- Empowering users
  - "…the ability to define one's own abstractions, or at least establish the specificity of self-description, rather than being subjected to the abstractions of others. " [p7]
  - User-constructed abstractions
    - User carries out direct manipulation actions
    - Inference algorithm recognizes repeated pattern and constructs abstraction

References

- [Kramer 2007] Kramer, J., *Is abstraction the key to computing?* Communications of the ACM, 2007. **50**(4): p. 36-42.
- [BCG 2008] Blackwell, A.F., L. Church, and T. Green, *The Abstract is an Enemy: Alternative Perspectives to Computational Thinking*, in *Psychology of Programming Interest Group 20th Annual Workshop*. 2008: Lancaster, UK